July 2001
DTN-17, Revision 0-1

Active Monitor on the Sun Cobalt™ Sausalito Architecture

1 What is Active Monitor

Active Monitor (AM) is a utility that periodically monitors key subsystems of your Sun Cobalt™ Qube 3 appliance. While Active Monitor has existed on previous appliance products, it has been updated and enhanced for the Sausalito architecture. Sausalito is the software architecture first implemented on the Sun Cobalt Qube 3 appliance. In the Sausalito environment, Active Monitor is a harness for both module-provided and built-in system monitoring routines. AM uses the Sausalito event model that allows the core of the work to be common, with the details tailored to each module's needs.

For more information on Sausalito, see the *Sausalito Developer's Guide*; see:
`ftp://ftp.cobalt.com/pub/developer/TechNotes/SSDK.pdf`

Table of Contents

1.1 Audience

The audience for this technical note are Sun Cobalt developers who want to write applications that use Sausalito's Active Monitor, or system administrators that want to fine-tune Active Monitor's behavior.

1.2 Applicable Products

Active Monitor is included as part of the software payload on the Sun Cobalt Qube 3 servers.

2 The Active Monitor Process

Active Monitor is run periodically from the system `cron` daemon. The program that is run is called `swatch`, for *System Watch*. On startup, `swatch` queries the Cobalt Configuration Engine (CCE) for a list of AM tests. For each test configuration found, `swatch` runs the specified test, and checks the status of the subsystem or service. If a problem is detected, `swatch` attempts to rectify the situation. The action it takes depends on the type of test run.

When the test is done, the current state of the service is stored in CCE. Monitored entities can have one of four states:

- GREEN (good)
- YELLOW (trouble)
- RED (severe trouble)
- WHITE (no information available or not monitored)

Because the AM state information is stored in CCE, the standard method of registering handlers for events can be used to perform actions when AM detects problems. In addition to the normal event facilities, `swatch` sends email containing all the changed messages to the system administrator, summarizing any changes it made.

3 Defining Monitored Entities with Schemas

Every test run by `swatch` is defined by a namespace on the Active Monitor object within CCE. Every Sausalito system should have exactly one `ActiveMonitor` object. In order for a namespace to be considered a test definition, it must have a minimum set of properties defined, which tells `swatch` how to handle the test.

The following schema class definition is the minimum set of required properties. Additional properties are allowed, and in some cases, required. You must specify the desired default values for each property in the schema.

```
<class name="ActiveMonitor" namespace="Template" version="1.0">
    <property name="enabled"            type="boolean"  default="1"/>
    <property name="monitor"            type="boolean"  default="1"/>

    <property name="type"              type="scalar"   default="exec"/>
    <property name="typeData"           type="scalar"
        default="/path/to/an/executable"/>

    <property name="currentState"       type="amstate"  default="N"/>
    <property name="currentMessage"     type="scalar"   default=""/>
    <property name="lastChange"         type="uint"     default="0"/>
    <property name="lastRun"            type="uint"     default="0"/>

    <property name="nameTag"            type="scalar"
        default="[[base-template.amName]]"/>
</class>
```

Table 1 lists the Active Monitor required properties.

Table 1 Active Monitor Required Properties

Property	Description
enabled	Boolean flag - is this service enabled?
monitor	Boolean flag - are we monitoring this service?

Table 1 Active Monitor Required Properties

Property	Description
type	the type of test
typeData	type-specific data about the test
currentState	the current state of this service (N, G, Y, or R)
lastChange	timestamp of last state change
lastRun	timestamp of last test run
currentMessage	the current message to display (internationalizable)
nameTag	the string to display in the UI for this service (internationalizable)

Along with required properties, there are some globally optional properties:

```
<property name="URL"     type="scalar"  default="/path/to/some.php"/>
<property name="UIGroup" type="scalar"  default="system"/>
<property name="hideUI"  type="boolean" default=""/>
```

Table 2 lists Active Monitor optional properties.

Table 2 Active Monitor Optional Properties

Property	Description
URL	a web-page with details about this service
UIGroup	defines which block of the UI this service is in (system, service, or other)
hideUI	boolean flag - should the UI hide this item?

4 Tests and Test Types

To make AM tests as flexible as possible, several built-in test types are provided, as well as an external test type. The following are valid test types:

- TCP
- UDP
- EXEC
- AGGREGATE

TCP and UDP are built-in tests, EXEC is an external test, and AGGREGATE is a wrapper for other tests.

4.1 TCP/UDP Tests

The TCP and UDP style tests are very elementary tests. A TCP or UDP connection is made to the port number specified in the `typeData` property. If the connection succeeds, the current state is set to green (G) and the appropriate message is stored. If the connection fails, the command listed in the `restart` property is run, and the test is retried. This will happen up to the number of times specified by the `retries` property. If, after retrying the appropriate number of times, the connection can not be made, the current state is set to red (R).

The messages for the green and red states are also stored in properties: `greenMsg` and `redMsg` respectively. If these are not specified in the schema, AM issues default messages. The following are required properties for TCP/UDP tests:

```
<property name="restart"  type="scalar"  default="/path/to/restart"/>
<property name="retries"  type="int"     default="3"/>
<property name="greenMsg" type="scalar"  default="Green Message"/>
<property name="redMsg"   type="scalar"  default="Red Message"/>
```

4.2 EXEC Tests

The EXEC test is the most flexible of AM tests. The program specified in the `typeData` property is executed, and its return value indicates the new current state. Before being executed, every property in the schema is stored as an environment variable. This allows the executing test to query things like thresholds or other data. Any property name that is not used for other purposes may be added to the schema to hold test-specific variables.

The current state is set by the exit value from the executing process. The exit values can be read symbolically from the file `/usr/sausalito/swatch/statecodes`. The following are allowed exit values, and their meanings:

- AM_STATE_NOINFO - no information to report
- AM_STATE_GREEN - status is green
- AM_STATE_YELLOW - status is yellow
- AM_STATE_RED - status is red
- AM_STATE_NOCHANGE - do not change the previous status

For ease of use, you can source this file from a Bourne shell (like *bash*). For additional flexibility, the Perl module `AM::Util` (located in `/usr/sausalito/perl`) provides the `am_get_statecodes()` function.

The last thing to note about EXEC tests is the current message property. Any text sent to standard output of the executed test becomes the current message.

4.3 AGGREGATE Tests

AGGREGATE tests are provided as a wrapper for other tests. An AGGREGATE test can contain any number of other tests, but can not contain another aggregate. The `typeData` property of the schema specifies a space-delimited list of namespaces which are members of the AGGREGATE. To prevent these namespaces from being run twice, they provide a Boolean true value in the `aggMember` property. The current state of an AGGREGATE is always the most severe state of its active members. The current message of an AGGREGATE is determined in the same fashion as TCP and UDP tests: from properties. The following are required or optional properties for aggregate tests or aggregate members:

```
<property name="aggMember"  type="boolean"  default="1"/>
<property name="greenMsg" type="scalar"  default="Green Message"/>
<property name="yellowMsg" type="scalar"  default="Yellow Message"/>
<property name="redMsg"   type="scalar"  default="Red Message"/>
```

5 Tweaking Active Monitor

While most services come with reasonable defaults, some times it is desirable to tweak the behavior just a bit. Some support for this is provided through the UI, and some is not. To access the hidden functionality, it is necessary to use the command line tool `cceclient` and the CSCP protocol directly. You must either be `root`, or use the `CSCP AUTH` command to become the Admin to CCE.

```
/usr/sausalito/bin/cceclient

< 100 CSCP/0.76
< 200 READY

> auth admin adminpass

< 109 SESSIONID UINOiVOgwiYwvrCAGoWciiZK6k7xWtpF10j7LVd531sohSOL4y
< 201 OK
```

First, find the `ActiveMonitor` object:

```
> find ActiveMonitor

< 104 OBJECT 12
< 201 OK
```

In this instance, the `ActiveMonitor` object ID is 12.

Find the list of namespaces:

```
> names 12

< 102 NAMESPACE CPU
< 102 NAMESPACE Memory
< 201 OK
```

Look at the current state of a test:

```
> get 12.CPU

< 102 DATA NAMESPACE = "CPU"
< 102 DATA CLASSVER = "1.2"
< 102 DATA UIGroup = "system"
< 102 DATA lastChange = "984532577"
< 102 DATA currentState = "N"
< 102 DATA type = "exec"
< 102 DATA yThreshhold = "4.0"
< 102 DATA typeData = "/usr/sausalito/swatch/bin/am_cpu.sh"
< 102 DATA rThreshhold = "6.0"
< 102 DATA currentMessage = ""
< 102 DATA lastRun = "970635928"
< 102 DATA enabled = "1"
```

```
< 102 DATA nameTag = "[[base-am.amCPUName]]"
< 102 DATA monitor = "1"
< 102 DATA URL = "/base/am/cpu_details.php"
< 201 OK
```

Now, you can change the properties, and make them more sensitive. Type:

```
> set 12.CPU yThreshhold=1 rThreshhold=2

< 201 OK
```

The next time `swatch` runs, the CPU monitoring program finds the new thresholds, and uses them.